**1. Введение**

**Название продукта:** Web-Library

**Контекст:** Web-Library — это веб-приложение, реализованное на Java Spring Boot, которое функционирует как цифровая библиотека. Продукт позволяет аутентифицированным пользователям просматривать каталог книг, управлять своей личной подборкой понравившихся книг, а также просматривать и оставлять рецензии. Администраторы системы обладают полными правами на управление всеми сущностями системы: книгами, авторами, жанрами, рецензиями и пользователями через интегрированную административную панель.

**Границы проекта:**

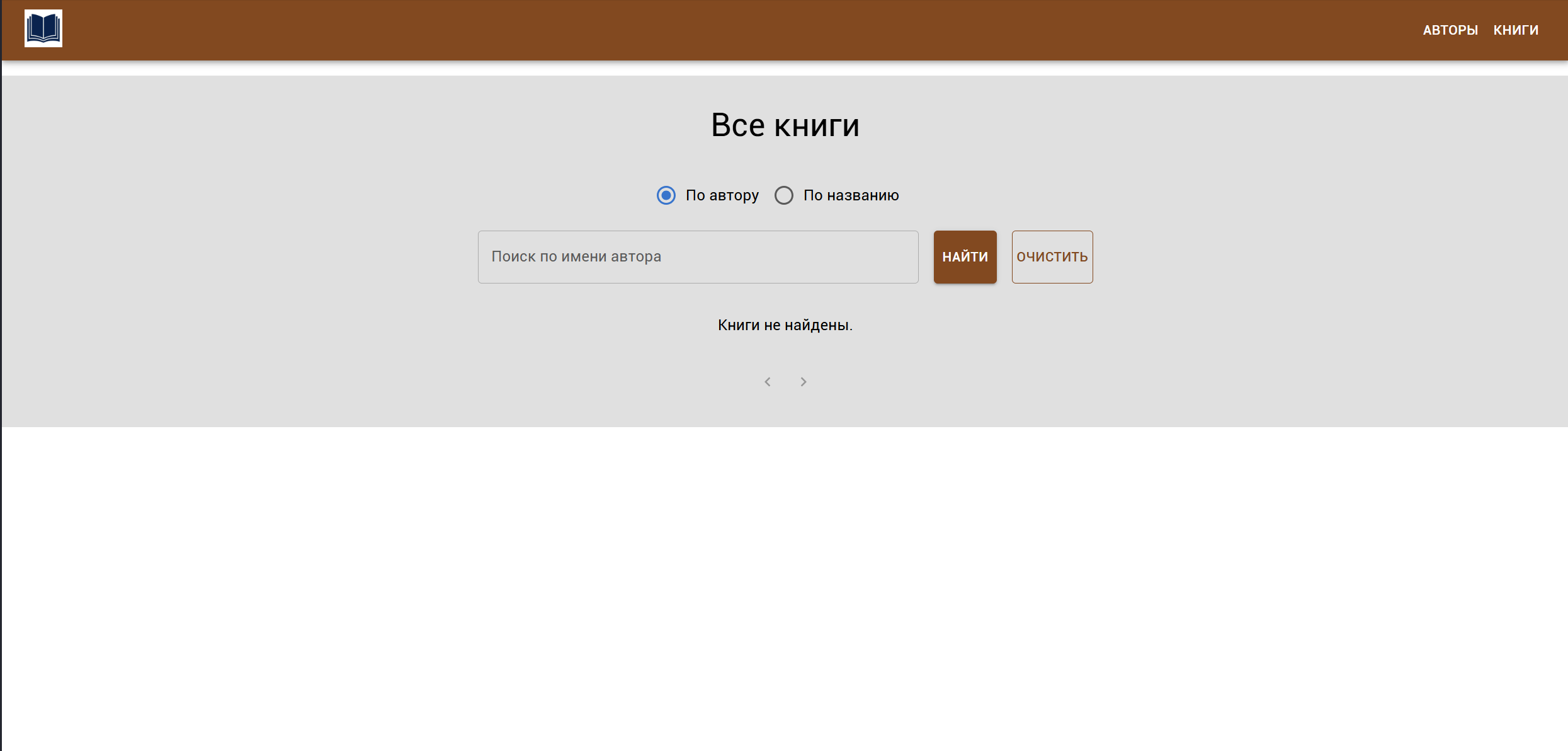
* Продукт **будет** предоставлять функционал аутентификации и авторизации пользователей.
* Продукт **будет** предоставлять функционал для добавления книг в личный список "Понравившиеся".
* Продукт **не будет** предоставлять функционал для чтения полного текста книг онлайн (текст книги не хранится и не отображается, согласно коду).
* Продукт **не будет** реализовывать сложную систему ролей (реализованы базовые роли "USER" и "ADMIN").
* Продукт **не будет** реализовывать систему бронирования или скачивания физических книг.

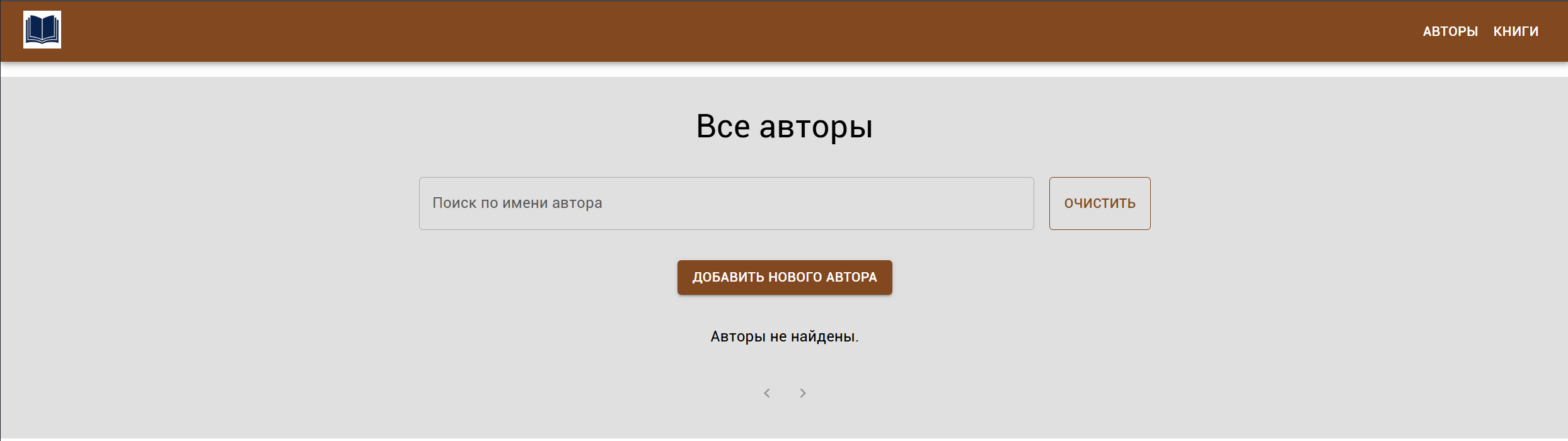
**2. Требования пользователя**

**2.1. Программные интерфейсы**

* **Backend:** Java 17, фреймворк Spring Boot 3.1.4.
* **База данных:** Реляционная СУБД (на основе использования Spring Data JPA, Hibernate и сущностей с аннотациями @Entity).
* **Безопасность:** Spring Security для аутентификации, авторизации и защиты от CSRF-атак.
* **Шаблонизатор:** Thymeleaf для рендеринга HTML-страниц на стороне сервера.
* **Валидация:** Bean Validation (аннотации @NotEmpty, @Size и т.д.).
* **Тестирование:** JUnit, Mockito (судя по наличию тестовых классов).

**2.2. Интерфейс пользователя**





**Текстовое описание (на основе анализа контроллеров и шаблонов):**  
Интерфейс состоит из HTML-страниц, генерируемых на сервере. Основные страницы:

1. **Страницы аутентификации:** /auth/login, /auth/registration.
2. **Главная страница:** /library — отображает каталог книг.
3. **Страница книги:** /library/{id} — отображает подробную информацию о книге, кнопку "Like", список рецензий и форму для добавления новой рецензии.
4. **Страница профиля пользователя:** /user/profile — отображает информацию о пользователе и его список понравившихся книг.
5. **Административная панель:** Доступна по различным endpoint'ам (например, /admin/books, /admin/people). Предоставляет интерфейс для просмотра, создания, редактирования и удаления записей о книгах, авторах, жанрах, рецензиях и пользователях.

**Таблица "Действие-Реакция":**

|  |  |
| --- | --- |
| Действие пользователя | Реакция системы (на основе кода) |
| Пользователь нажимает "Like" на странице книги. | Система проверяет авторизацию. Книга добавляется в связующую таблицу liked\_books. Обновляется статус кнопки. |
| Пользователь отправляет рецензию. | Система валидирует текст рецензии, сохраняет ее в БД, привязывая к текущему пользователю и книге. Обновляет список рецензий на странице. |
| Администратор редактирует книгу через форму /admin/books/{id}/edit. | Система принимает PATCH-запрос, валидирует данные, обновляет сущность Book в БД и перенаправляет на страницу администратора. |

**2.3. Характеристики пользователей**

|  |  |
| --- | --- |
| Группа пользователей | Характеристики (на основе кода) |
| **Анонимный пользователь** (permitAll() на /auth/\*\*, /library). | Может просматривать каталог книг и страницы регистрации/входа. Не может оставлять рецензии или лайки. |
| **Аутентифицированный пользователь** (Роль: USER). | Может просматривать каталог, оставлять рецензии, добавлять/удалять книги в раздел "Понравившиеся" в своем профиле. |
| **Администратор** (Роль: ADMIN, доступ к /admin/\*\*). | Имеет полный доступ CRUD ко всем данным в системе (управление книгами, авторами, жанрами, рецензиями и пользователями). |

**2.4. Предположения и зависимости**

* Проект зависит от экосистемы Spring Boot и корректной настройки базы данных.
* Предполагается, что вся бизнес-логика выполняется на стороне сервера (не SPA).
* Интерфейс администратора предназначен для технически подкованных пользователей.
* Функционал "лайков" и рецензий жестко привязан к авторизованным сессиям пользователей.

**3. Системные требования**

**3.1. Функциональные требования**

1. **FR-01: Управление аутентификацией и авторизацией**
   * **FR-01.01:** Система должна предоставлять возможность регистрации нового пользователя (эндпоинт /auth/registration, метод POST).
   * **FR-01.02:** Система должна предоставлять возможность аутентификации существующего пользователя (эндпоинт /auth/login, метод POST).
   * **FR-01.03:** Система должна разграничивать доступ к эндпоинтам на основе ролей пользователя (USER, ADMIN).
2. **FR-02: Просмотр и управление каталогом книг**
   * **FR-02.01:** Система должна предоставлять всем пользователям возможность просматривать список книг (эндпоинт /library, метод GET).
   * **FR-02.02:** Система должна предоставлять всем пользователям возможность просматривать детальную информацию о конкретной книге (эндпоинт /library/{id}, метод GET).
3. **FR-03: Функционал пользователя**
   * **FR-03.01:** Система должна позволять аутентифицированному пользователю добавлять книгу в список "Понравившиеся" и удалять ее оттуда.
   * **FR-03.02:** Система должна позволять аутентифицированному пользователю создавать рецензии на книги (эндпоинт /reviews/create, метод POST).
   * **FR-03.03:** Система должна предоставлять аутентифицированному пользователю возможность просмотреть свой профиль со списком понравившихся книг (эндпоинт /user/profile, метод GET).
4. **FR-04: Административное управление (CRUD)**
   * **FR-04.01:** Система должна предоставлять администратору возможность создавать, просматривать, обновлять и удалять сущности Book (эндпоинты /admin/books/\*\*).
   * **FR-04.02:** Система должна предоставлять администратору возможность создавать, просматривать, обновлять и удалять сущности Person (пользователя) (эндпоинты /admin/people/\*\*).
   * **FR-04.03:** Система должна предоставлять администратору возможность управлять сущностями Author и Genre.

**3.2. Нефункциональные требования**

**3.2.1. АТРИБУТЫ КАЧЕСТВА**

* **Безопасность (Security):** Критически важен. Реализовано с помощью Spring Security.
  + **Важность:** Для защиты пользовательских данных и разграничения прав доступа.
  + **Метрика:** Отсутствие уязвимостей (например, возможность неавторизованного доступа к /admin/\*\*, CSRF-атак на формы редактирования). Проверяется код-ревью и пентест-инструментами.
* **Надежность (Reliability):** Важен.
  + **Важность:** Система должна consistently обрабатывать запросы без неожиданных падений.
  + **Метрика:** Время безотказной работы (uptime). Обработка исключений (например, попытка доступа к несуществующей книге /library/99999 не должна приводить к 500 error, а обрабатываться корректно).
* **Удобство сопровождения (Maintainability):** Высокая важность.
  + **Важность:** Код хорошо структурирован (пакеты controller, service, repository, entity), что упрощает его дальнейшее развитие и исправление ошибок.
  + **Метрика:** Наличие модульных тестов (как в вашем проекте) для ключевых сервисов, что снижает риски при внесении изменений.
* **Производительность (Performance):** Средняя важность на текущем этапе.
  + **Важность:** Страницы должны загружаться за приемлемое время.
  + **Метрика:** Время отклика сервера на ключевые запросы (например, загрузка главной страницы, профиля пользователя). В будущем может потребоваться пагинация для списка книг.